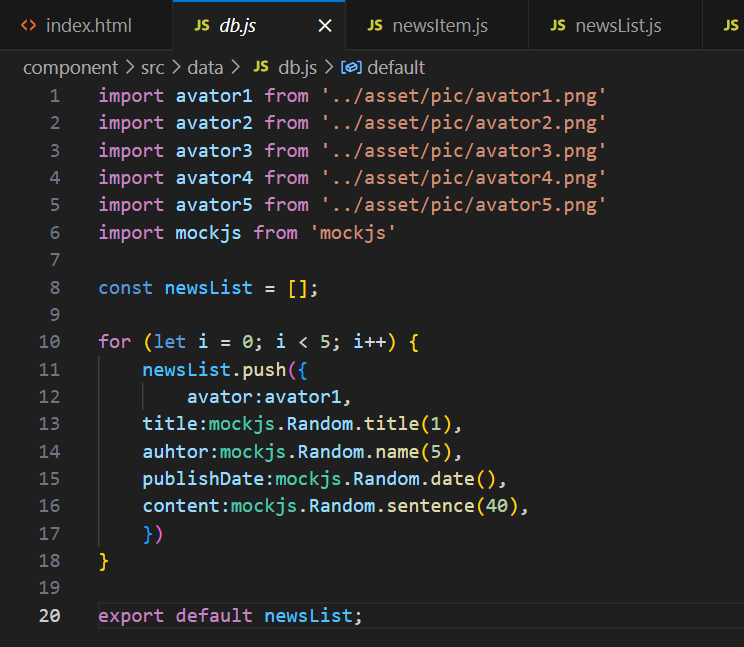
在正式介绍如何实现一个ui组件之前，我展示一下，具体上来说我们需要做一个什么东西。



我们就是简单的模拟微博手机端，或者说facebook手机端那种效果。

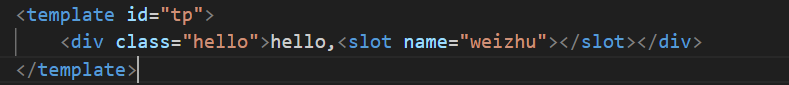
我们先来使用mockJs来生成我们需要的信息，在真实的开发中，这个数据一般都是从后台通过api抓过来的。



在正式介绍如何实现如何自定一个ui组件之前，我们还需要知道webcomponent中一些比较重要的生命周期hooks函数和api，template标签、还有一个相当重要的标签”slot”。

什么是webComponent中的template和slot标签。

像之前我们定义shadowDOM的结构都是在customElement的构造函数中定义的，但是这样可能不是很方便，而且如果我们都使用document的api来完成shadowDOM元素的定义的话，需要对Js的dom操作有很深的理解。所以它给我们提供了一个template标签，我们可以直接在这个标签中写dom结构。但这个东西最终是不会被渲染到页面上的。

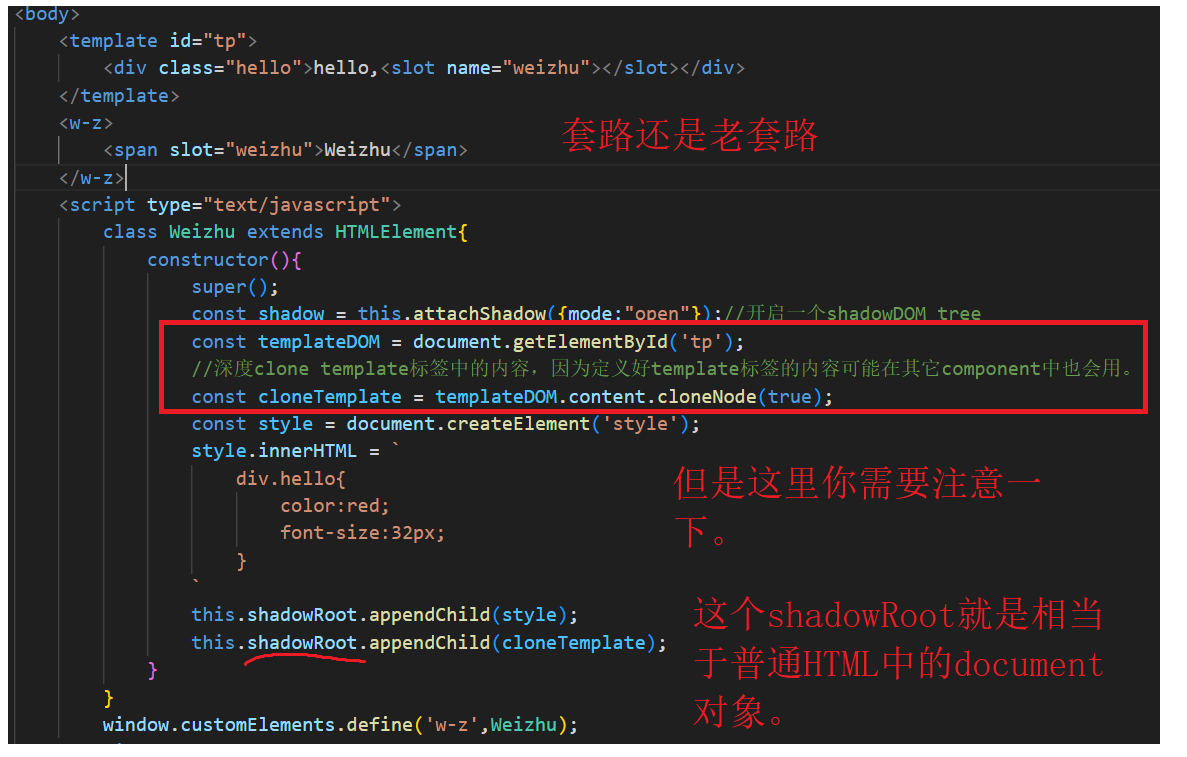


Slot标签就是你定义一个插槽，类似于vue中的那种操作，在最终使用的时候,它会给你自动替换成对应的内容。



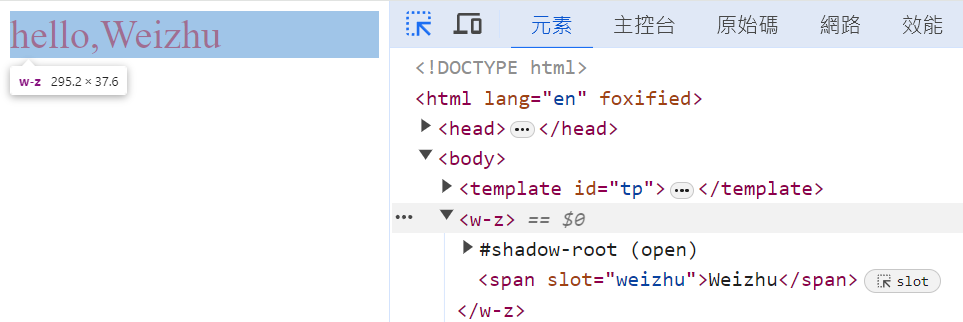
这个template就是为了创建customElement的模板用的，有关它的使用细节，可以去mdn上看文档，不难。

最后我们来看一下他们两个具体上是如何配合起来使用的。



其中这个shadowRoot,就是shadowDom Tree的根元素对象，就类似于html中的document对象。

最后我们来看一下页面效果（非常完美）：



customElement构造函数中几个比较重要的生命周期钩子函数

1. construtor --------在customElement构建的时候触发
2. connectedCallback --------在shadowDOM被真正添加到真实的DOM结构中触发。
3. attributeChangedCallback -----shadowhost标签上的属性值被改变的时候触发，注意属性被创建的时候也会触发。
4. adoptedCallback/disconnectedCallback----------adoptedCallbak是指customElement被移动到其它HTML DOM结构中的时候触发（你会好奇，为什么一个HTML文件中怎么可能会有两个HTML DOM结构呢，可以通过fragement标签 实现），disconnectedCallback则是在customElement被卸载，也就是这个节点被删除的时候触发。

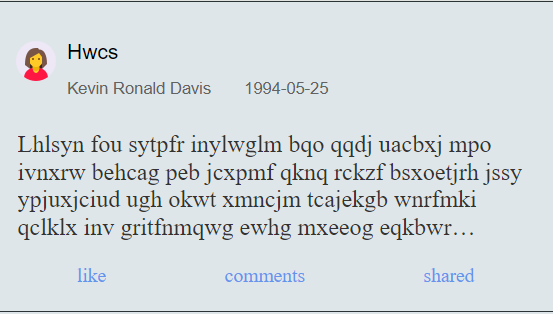
具体它们的使用方式，mdn上有很详细的介绍，这里如果我们逐个演示的话，会把篇幅弄得很大，不易阅读。但是我需要提醒的是，在之前我们都是在constructor中构建我们的customElement，在mdn上，它建议我们在connectedCallback中完成这一部分的工作。

有关template中slot元素的几个比较重要的api

1. HTMLSlotElement.assign() ----用于在customElement中给slot标签分配具体要替换的内容的。
2. HTMLSlotElement.assignedElements------用于获取所有被分配到当前slot标签的内容。

手写一个component组件

在开发中，我们基本上都是需要先完成粒度比较小的东西的开发与测试，这里我们也是一样先完成单独的每一个newsItem组件的开发。

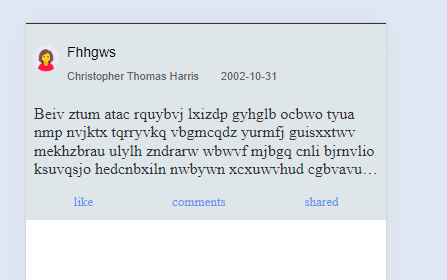


其中的css布局啥的，我这里就不说了，很简单的东西。我们直接来看怎么构建这个粒度较小的组件。



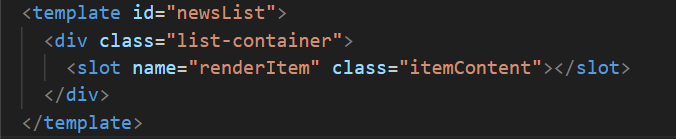
我们重点来看看attributeChangedCallback生命周期hook函数的使用。首先你一定要使用observedAttributes静态方法设置你要监听什么属性。然后我们就可以通过监听这个属性值的形式实现数据的传输。但是由于属性值只能是字符串，所以我们要用JSON对象转化一下。

我们来看一下效果：



相当完美！

紧接着我们就可以构建newsList这个customElement了。

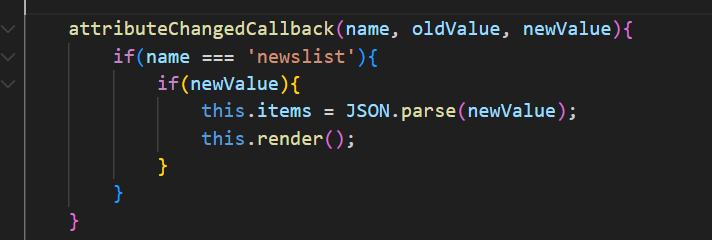


这个newslist的构建重点在监听它属性的变动，以及获取它内部的slot标签。



我们一个个来看：

首先我们来看比较简单的，我们仍旧是需要通过监听属性来完成数据的接收。



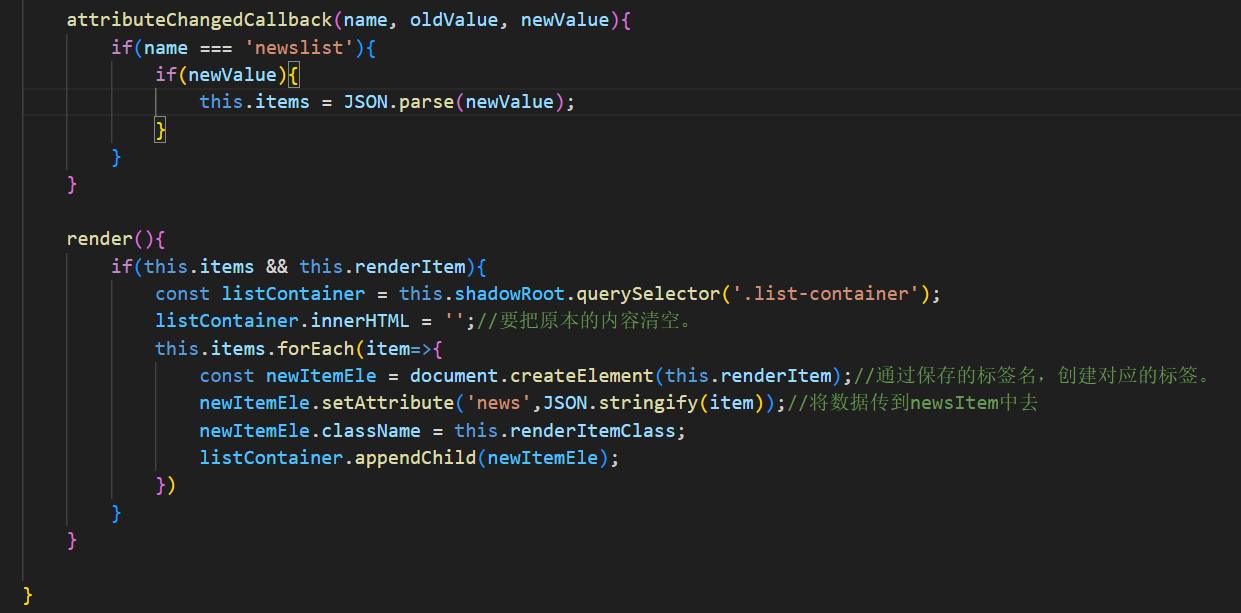
紧接着我们来看看元素的构建：



大致的逻辑是，我们需要先将slot标签给读取出来，然后获取它身上的className。这个className其实最后是用来定义newsitem的样式用的，没有什么特殊用途。

紧接着我们给slot对象添加一个slotchange事件，这个事件会在slot被移除或添加的时候触发，也可以是它的属性被修改或者删除的时候触发。我们这里是移除，因为我们要在newslist中渲染newsItem。在这个事件处理函数中，我们需要将最初分配给slot的内容获取下来，然后保留一些关键信息。之所以在connectedCallback中也执行一次render()方法，原因是我一开始没搞清它跟attributechangedCallback那个先触发。（后面经过测试，发现是attribute先触发，所以我们还是在connectedCallback中完成数据的渲染。）

剩下的部分其实就非常简单了，这里我只把代码贴出来，具体的我就不解释了。



我们来看一下最终渲染的效果：



非常完美！